Lab 2: Experimenting with Symbolic Execution and Fuzzing

Objective

The objective of this lab is to experiment with symbolic execution and fuzzing tools on programs with inserted vulnerabilities and observe how well they can discover these vulnerabilities.

Challenge Programs

Pick one program (except CADET_00001 and CADET_new) from these DARPA CGC challenge programs: https://github.com/hengyin/cb-multios/tree/master/challenges.

Tools

You are expected to experiment with the following tools:

Klee: https://github.com/klee/klee

IJON: https://github.com/RUB-SysSec/ijon

Read the documentation for each tool. For Klee, you can directly use its container image from docker hub: https://hub.docker.com/r/klee/klee. For IJON, you must build the tool by following its instructions.

Task 1: Klee (40%)

Use Klee on the program you choose, answer the following questions: 1) Can Klee find inputs that crash it? and 2) How much code coverage can Klee reach for it?

Please also provide your own explanation for the answers you have, such as why Klee can or cannot crash it, and why Klee can reach very high or low code coverage.

Task 2: AFL (30%)

IJON without annotations is essential an AFL. Use AFL on the program you choose, and answer the following questions: 1) Can AFL find inputs that crash it? and 2) How much code coverage can AFL reach for it?

Please also provide your own explanation for the answers you have, such as why AFL can or cannot crash it, and why AFL can reach very high or low code coverage.

Task 3: IJON (30%)

Add IJON annotations on the program you choose, and answer the following questions: 1) explain where you add annotations and why you add them; 2) Compare the difference before and after you add these annotations in terms of crashes and code coverage.

A few important points:

1. The report should include figures and screenshots for important steps and results.
2. You are not expected to run each experiment for very long. Each experiment can be as short as five to ten minutes.